Journal of Intelligent and Connected Vehicles
2024, 7(2): 86–96
https://doi.org/10.26599/JICV.2023.9210026

Research Article

Online learning-based model predictive trajectory control for connected and autonomous vehicles: Modeling and physical tests

Qianwen Li[1], Peng Zhang[2], Handong Yao[1], Zhiwei Chen[3], Xiaopeng Li[2,✉]

[1]*School of Environmental, Civil, Agricultural and Mechanical Engineering, University of Georgia, Athens 30602, USA*
[2]*Department of Civil and Environmental Engineering, University of Wisconsin–Madison, Madison 53706, USA*
[3]*Department of Civil, Architectural, and Environmental Engineering, Drexel University, Philadelphia 19104, USA*

Received: October 4, 2023; Revised: October 26, 2023; Accepted: November 9, 2023

ABSTRACT: Motivated by the promising benefits of connected and autonomous vehicles (CAVs) in improving fuel efficiency, mitigating congestion, and enhancing safety, numerous theoretical models have been proposed to plan CAV multiple-step trajectories (time–specific speed/location trajectories) to accomplish various operations. However, limited efforts have been made to develop proper trajectory control techniques to regulate vehicle movements to follow multiple-step trajectories and test the performance of theoretical trajectory planning models with field experiments. Without an effective control method, the benefits of theoretical models for CAV trajectory planning can be difficult to harvest. This study proposes an online learning-based model predictive vehicle trajectory control structure to follow time–specific speed and location profiles. Unlike single-step controllers that are dominantly used in the literature, a multiple-step model predictive controller is adopted to control the vehicle's longitudinal movements for higher accuracy. The model predictive controller output (speed) cannot be interpreted by vehicles. A reinforcement learning agent is used to convert the speed value to the vehicle's direct control variable (i.e., throttle/brake). The reinforcement learning agent captures real-time changes in the operating environment. This is valuable in saving parameter calibration resources and improving trajectory control accuracy. A line tracking controller keeps vehicles on track. The proposed control structure is tested using reduced-scale robot cars. The adaptivity of the proposed control structure is demonstrated by changing the vehicle load. Then, experiments on two fundamental CAV platoon operations (i.e., platooning and split) show the effectiveness of the proposed trajectory control structure in regulating robot movements to follow time–specific reference trajectories.

KEYWORDS: connected and autonomous vehicles (CAVs), reinforcement learning, physical tests, time–specific speed and location, longitudinal and lateral control

1 Introduction

Connected and autonomous vehicles (CAVs) have witnessed remarkable development, given their potential to improve fuel efficiency, mitigate congestion, and enhance safety (Li et al., 2022; Li and Li, 2023). Enabled by vehicle automation technology, CAVs' trajectories can be precisely designed to accomplish various operations (e.g., passing a signalized intersection and forming a platoon) with specific objectives (e.g., maximizing mobility and minimizing fuel consumption) (Li and Yao, 2022; Li et al., 2021).

In the past decades, extensive models have been proposed to plan CAV trajectories in various application contexts, including but not limited to signalized intersections, stop-controlled intersections, ramp merging, platooning, and speed harmonization (Yao et al., 2018). Existing trajectory planning models are generally single-step or multiple-step. Single-step trajectory planning designs vehicle movement step by step based on a set of linear/nonlinear equations (Raboy et al., 2021). The output is usually a single speed/acceleration value for the current or next time step. In contrast, multiple-step trajectory planning devises a time series of trajectories, including speed/acceleration/location values in multiple future time steps (He et al., 2015).

Although fruitful theoretical efforts have been made on CAV trajectory planning, physical experiments to validate the theoretical models are relatively scarce, which are necessary for transferring theoretical models into implementable technology in the real world. Most of the existing trajectory planning models are evaluated with simulations (Li and Yao, 2022; Yao and Li, 2020), which reveal limited insights into the model performance in real-world settings. The fundamental issue is that in simulations, CAV motions are often assumed to be precisely controllable, i.e., a CAV (or a group of CAVs) can exactly follow the planned trajectory (trajectories) with zero location and speed errors. However, in reality, precise trajectory control is almost impossible for the following reasons. First, the mechanical control of a vehicle may have response lag and inaccuracy depending on its electric control and powertrain mechanisms (Su et al., 2018). Further, vehicle operations are subject to various exogenous disturbances, e.g., weather, roadway condition, and vehicle load. These disturbances cause unpredicted errors between the planned trajectories and actual trajectories. Optimal trajectory control strategies are needed to reduce the error between the planned and actual trajectories as much as possible.

Controlling a vehicle to follow single-step trajectories is relatively easy since only one specific value (e.g.,

✉ Corresponding author.
E-mail: xli2485@wisc.edu

speed/acceleration) needs to be followed. The control of multiple-step trajectories is more challenging. Vehicles must be controlled to follow a time series of locations, during which accumulated errors must be considered. For example, if a vehicle misses some distance initially, it must be controlled to catch that distance. Otherwise, the vehicle cannot finish the operation with the expected performance, e.g., fuel efficiency and mobility, and even worse, consecutive vehicles may collide. Single-step controllers are the most commonly used optimal controllers (e.g., proportional–integral–derivative, PID). They have been widely used in field tests with single-step trajectory planning. The control input and output are single values (Milanes et al., 2010). For example, a controller can take the speed error as the input and generate the adjusted acceleration. Given its simplicity, single-step controllers have also been used with multiple-step trajectory planning (Ma et al., 2019, 2020). Although trajectories are planned for multiple future time steps, trajectory control still regulates the vehicle's single-step movement based on the current state. While single-step controllers are computationally efficient, the control performance cannot be guaranteed because it only focuses on the current step without planning for the future.

In comparison, model predictive control (MPC) requires a time series of the reference input and generates a time series of the system output. MPC has been used for vehicle path tracking (Tang et al., 2020). The objective is to ensure vehicles operate on a designated path (usually a two-dimensional curve). Vehicle longitudinal speed is usually not regulated/kept constant during path tracking. As a result, no time–specific location profiles are followed. A time–specific location profile is a time series of a vehicle's locations in a certain time horizon. Model predictive control has also been used for regulating a single vehicle's movements to follow a time–specific speed profile, i.e., a time series of a vehicle's speeds in a certain time horizon. The objective is to control the vehicle's speed as close to the reference speed as possible. Since no surrounding traffic is considered, vehicle time–specific locations are not regulated. The accumulated location error is not addressed. Specifically, a vehicle does not need to mitigate the missed travel distance as long as the current speed is consistent with the reference. Without proper control of time–specific locations, existing model predictive control techniques cannot be applied to traffic streams where vehicles interact because consecutive vehicles may collide (Campion et al., 2018). However, using model predictive control to regulate vehicle movements to follow time–specific location trajectories has not received much attention in the CAV literature, making it difficult to harvest the promising benefits of the CAV technology.

More importantly, the existing trajectory controllers in field experiments usually output acceleration and speed, which cannot be interpreted by vehicles and robots (Morales and Nijmeijer, 2016). The direct control variable is throttle/brake for vehicles and motor rotation per minute for robots. The major challenge is how to convert the controller output into the direct control variable of a vehicle/robot. A few existing studies have attempted to approach this challenge. Rajamani et al. (2000) calculated the throttle/brake angle using a sliding surface controller. Milanes et al. (2010) calibrated a static lookup table to capture the relationship between the gas pedal and the control error (speed and distance errors). Recently, some studies have used PID controllers to convert speed/acceleration instructions to throttle/brake (Raboy et al., 2021). For accurate speed control, different model parameters should be calibrated in different operating environments (i.e., the combination of various factors, including but not limited to roadway conditions, vehicle load, and weather). It is impossible to enumerate all possible operating environments. As a result, a specific set of parameters will be used in the environment that it is obtained and similar environments with certain slight changes. For example, parameters developed for vehicles operating on a flat roadway segment can be used when vehicles are driving on segments with a small slope. This way, the control accuracy is not guaranteed. Intuitively, calibrating more parameters in more environments contributes to higher control accuracy. However, the calibration usually requires a significant amount of resources (e.g., human, time, and equipment). An adaptive conversion from the controller output to the vehicle/robot control variable that captures real-time environment changes is demanded to improve vehicle control performance and save parameter calibration resources.

Motivated by these research gaps, this study proposes an online learning-based model predictive trajectory control structure to regulate vehicle movements to follow time–specific trajectories generated by multiple-step planning models. A model predictive controller is chosen for longitudinal speed control for higher accuracy instead of dominantly used single-step controllers in the literature. A reinforcement learning agent constructs and maintains the dynamic conversion between the speed controller output and the vehicle/robot direct control variable. In addition, a lateral controller is used to control the vehicle's orientation such that it stays on track. With the proposed control structure, accurate CAV speed and location control is feasible. As a result, the promising benefits (e.g., fuel saving and mobility improvement) of various CAV operations (e.g., platooning and eco-driving) can be harvested the most. Reduced-scale robot cars are utilized in this study because they require fewer resources and do not impose any safety concerns.

The remainder of this paper is organized as follows. Section 2 first describes the investigated trajectory control problem and then introduces the optimal trajectory control structure. Section 3 describes the reduced-scale platform. Section 4 presents field experiment results and verifies the effectiveness of the proposed control structure. Section 5 concludes this paper and points out future research directions.

2 Optimal control

This section first introduces the investigated trajectory control problem. Next, it proposes an online learning-based model predictive control structure to regulate vehicle longitudinal and lateral movements to follow time–specific speed and location profiles. Further, a benchmark is developed for comparison.

2.1 Trajectory control problem statement

This study investigates the trajectory control problem to regulate CAV movements to finish the intended operation (e.g., speed harmonization and platooning) safely and achieve the expected performance (e.g., improving mobility and fuel efficiency).

CAV control includes two components, i.e., longitudinal control and lateral control. Longitudinal control is to regulate a vehicle's longitudinal movements to follow the time–specific speed and location trajectories (i.e., references). Lateral control is to regulate a vehicle's lateral motions to follow the time–specific orientation reference so that it can stay on the designated track.

Given a period $t \in [0, T]$, a time–specific location reference is the set of a vehicle's target locations $\widehat{\mathcal{X}} := \{\widehat{x}(t)\}_{t\in[0,T]}$. A time–specific speed reference is the set of a vehicle's target speeds $\widehat{\mathcal{V}} := \{\widehat{v}(t)\}_{t\in[0,T]}$. A time–specific orientation reference is the set of a vehicle's target orientation angles $\widehat{\mathcal{O}} := \{\widehat{o}(t)\}_{t\in[0,T]}$. Given

清華大学出版社 Tsinghua University Press | IEEE

the response lag and inaccuracy of vehicle mechanical control and the presence of various disturbances in the real world, e.g., road conditions, weather, and vehicle load, the actual movements of vehicles cannot be exactly like the references. A vehicle's actual location and speed profiles are denoted as $\mathcal{X} := \{x(t)\}_{t\in[0,T]}$ and $\mathcal{V} := \{v(t)\}_{t\in[0,T]}$. The actual orientation is denoted as $\mathcal{O} := \{o(t)\}_{t\in[0,T]}$. Errors are expected between the actual movements and the references, denoted by $e^x(t) := x(t) - \widehat{x}(t)$, $e^v(t) := v(t) - \widehat{v}(t)$, and $e^o(t) := o(t) - \widehat{o}(t)$. This study proposes an effective control structure to minimize these errors and maximize the corresponding benefits of actual CAV operations.

Vehicles interact with each other in real-world traffic. For safe operations (e.g., platooning), besides minimizing the individual vehicle control errors, the inter-vehicle distance should also be regulated so consecutive vehicles that do not collide. The gap between the preceding vehicle $n-1$ and the following vehicle n should always be no less than a safety distance, $g(t) := x_{n-1}(t) - x_n(t) \geq g$, to ensure the operation safety.

2.2 Online learning-based model predictive control

An online learning-based model predictive control (OLMPC) structure with three feedback loops is proposed to accomplish speed and location control while line tracking, as shown in Fig. 1. The longitudinal control outer loop regulates the vehicle to follow the target trajectory with a model predictive controller. The target trajectory ($\widehat{x}$ and $\widehat{v}$) and the vehicle's actual trajectory (x and v) are fed into the controller to generate the adjusted speed $\tilde{v}$. However, the adjusted speed cannot be interpreted by vehicles. Vehicles' direct control variable (DCV) is usually the throttle/brake. Thus, the longitudinal control inner loop converts the speed to the DCV with a reinforcement learning (RL) agent. The inputs of the RL agent are the adjusted speed $\tilde{v}$ and the vehicle's actual speed v. The output is the adjusted DCV. The RL agent dynamically updates the conversion to account for the vehicle operating environment changes. Finally, the lateral control loop controls the vehicle to follow a designated track with a tracking controller. The tracking error e^o is fed into the tracking controller to produce the adjusted orientation $\tilde{o}$. Once the adjusted DCV and orientation are applied, the vehicle's actual movement is measured after the vehicle dynamics and used for the next control step.

The frequency of the longitudinal inner loop should be greater than that of the outer loop so that the RL agent has enough time to compensate for disturbances before they affect the outer loop. The faster the inner loop is, the better the disturbance compensations are (Bolton, 2015). The resulting speed control accuracy would be higher. The remainder of this subsection presents the details of each control loop.

It should be noted that the proposed OLMPC is for individual vehicle control. Regarding traffic streams including multiple vehicles, each following vehicle should be equipped with an additional spacing/headway controller, which regulates the safe distance between vehicles so that no collisions happen, i.e., $x_{n-1}(t) - x_n \geq g$. Well-established control methods can achieve this goal, e.g., linear–quadratic regulator, PID, and MPC.

2.2.1 *Model predictive speed control*

A model predictive controller is used for the longitudinal outer loop to control the vehicle's longitudinal speed. It not only takes care of the current time step but also plans for the future. Thus, it is expected to yield better control accuracy than single-step controllers. The goal of the model predictive controller is to control vehicles to follow target trajectories ($\widehat{x}$ and $\widehat{v}$) smoothly for riding comfort and fuel efficiency. The objective function, Eq. (1), is formulated below to achieve this goal with a control period from t to $t+\Delta T$. The first two terms in the numerator are the trajectory control errors, and the last term regulates speed jumps.

$$J = \min_{v} \frac{\int_t^{t+\Delta T} w_x \times (x(t') - \widehat{x}(t'))^2 + w_v \times (v(t') - \widehat{v}(t'))^2 + w_a \times a(t')^2 \mathrm{d}t'}{\Delta T} \tag{1}$$

subject to Eq. (2):

$$\begin{cases} \underline{v} \leq v(t') \leq \overline{v} \\ \underline{a} \leq a(t') \leq \overline{a} \end{cases} \tag{2}$$

where $\widehat{v}(t')$ is the target speed; $v(t')$ is the vehicle's actual speed; $\widehat{x}(t') = \int_t^{t+t'} \widehat{v}(t^*)\,\mathrm{d}t^*$ is the target location; $x(t') = \int_t^{t+t'} v(t^*)\,\mathrm{d}t^*$ is the vehicle's actual location; $a(t') = \dfrac{\mathrm{d}v}{\mathrm{d}t}$ is the vehicle's actual acceleration; w_x, w_v, and w_a are the weight parameters. Control variable $v(t')$ is bounded by speed limits, i.e., $[\underline{v}, \overline{v}]$. The vehicle's acceleration is also bounded, i.e., $[\underline{a}, \overline{a}]$. State-

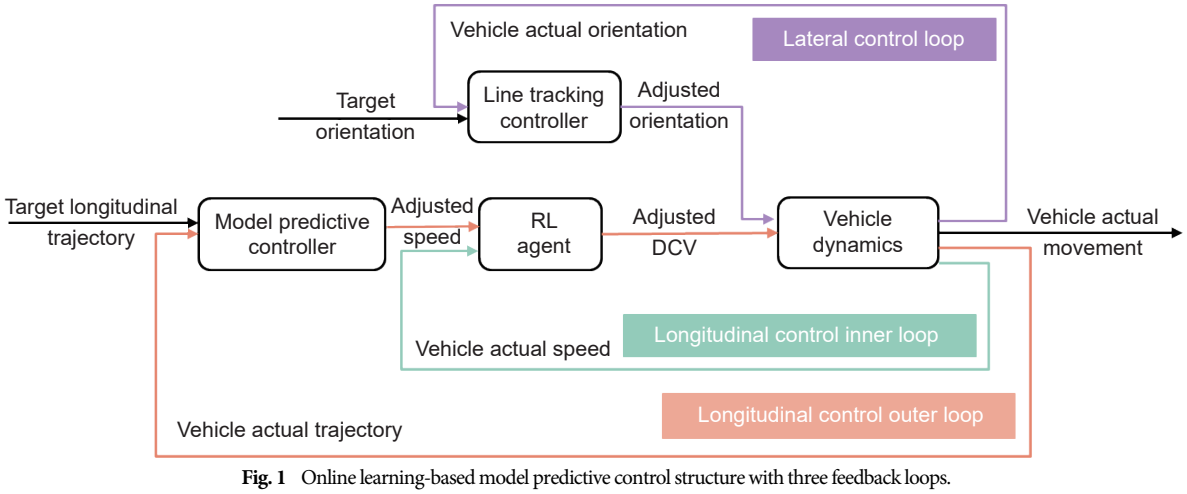

Fig. 1 Online learning-based model predictive control structure with three feedback loops.

of-the-art solvers can quickly solve this optimization problem after discretization with only dozens of variables. The resulting optimal solution is the adjusted speed sequence, denoted by $\tilde{\mathcal{V}} := \{\tilde{v}(\tau)\}_{\tau\in[t,t+\Delta T]}$, that regulates a vehicle's longitudinal movements to be close to the target.

2.2.2 Fuzzy Q-learning conversion

A Q-learning agent is chosen for the longitudinal control inner loop to convert the adjusted speed $\tilde{v}$ to the adjusted DCV so that vehicles can understand the control instruction, given its computation efficiency and satisfying performance demonstrated in the literature (Clifton and Laber, 2020; Yang et al., 2020). It updates the conversion online while the vehicle is running to accommodate possible changes in the operating environment, e.g., test track and vehicle load changes. The state of the Q-learning agent is defined as the longitudinal speed control error calculated as the difference between the vehicle's actual speed and the adjusted speed returned by the speed controller. This state is inherently continuous and unbounded. Consequently, the resulting Q-table, with states as rows and actions as columns, is of infinite size, rendering it impossible to be fully trained. Thus, the speed control error $e^v(t)$ is transformed in the range of $[-\varepsilon, \varepsilon]$ by a sigmoid function and is denoted by $e^u(t)$. Instead of simply discretizing $e^u(t)$ into a limited number of states by a unit, fuzzy logic is used to represent the continuous state space $e^u(t)$ with N state membership functions defined. Fuzzy logic can depict $e^u(t)$ with countless speed error statuses, which are the weighted summation of the N states. Parameters ε and N should be adjusted to produce satisfying conversion accuracy within a reasonable training time. Specifically, greater ε and N values depict the control error more precisely but result in a large q-table that can be difficult to converge. An example is given in Fig. 2 when $a = 0.6$ and $N = 7$. Speed control error (or state) membership functions include Negative Big (NB), Negative Medium (NM), Negative Small (NS), Zero (Z), Positive Small (PS), Positive Medium (PM), and Positive Big (PB) (Chiou et al., 2012). Note that these functions exhibit continuity throughout the speed control error e^u range. For instance, NB is positive when $e^u < -0.4$ and equal to zero when $e^u \geq -0.4$ (which may not be clearly discernible in Fig. 2 due to overlapping). For an error $e^u(t)$, the firing strength μ of each membership function is measured. μ indicates the degree to which the agent (i.e., vehicle) is in a state (i.e., speed control error). A set $\mathcal{M}$ is defined, including the indexes of membership functions whose firing strengths are greater than zero.

The action in Q-learning corresponds to the rate at which the vehicle's DCV is adjusted, and this parameter is also continuous and unbounded. Given that the DCV adjustment rate directly influences the vehicle's acceleration (with greater adjustment leading to higher acceleration), it is constrained within the range of $[-\sigma, \sigma]$ to avoid significant speed adjustment for riding comfort and fuel efficiency. Further, the adjusting rate is discretized by a small interval Δ, resulting J actions in total. The values of σ and Δ should be selected per application needs. A greater σ leads to a faster vehicle control response with the sacrifice of a greater speed adjustment. A greater Δ contributes to faster Q-learning convergence, but the speed control performance may not be as accurate because the adjustment is not precise. For every state $n \in [1, N]$, there are J actions available. The Q-learning agent, i.e, the vehicle, selects the optimal action that maximizes its

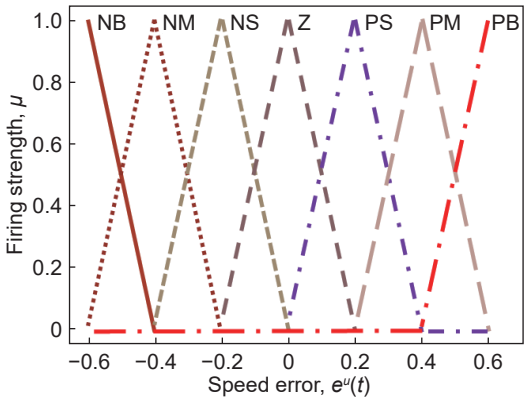

Fig. 2 State membership function when $a = 0.6$ and $N = 7$.

expected reward. This reward is instrumental in updating the q-table for guiding future action selections. The q-table is iteratively updated as the agent interacts with the environment, continuing until convergence is reached, signified by minimal changes in the table values.

An example is given as follows. Assume $e^u(t) = 0.5$, $m \in \mathcal{M} = \{6,7\}$ indicating that PM and PB are fired. The firing strength are equal, i.e., $\mu_6 = \mu_7 = 0.5$. Next, assume that the best action for PM is decreasing the DCV by 10%, and the best action for PB is decreasing the DCV by 8%. The DCV adjusting rate after fuzzy logic would be the sum of adjusting rates a_n weighted by firing strengths μ_n, i.e., $r^{\mathrm{DCV_{adj}}} = \sum_{n\in\mathcal{N}} a_n \times \mu_n = -0.10 \times 0.5 + (-0.08 \times 0.5) = -0.09$. The current DCV will be decreased by 9%. Subsequently, once the adjusted DCV is implemented, the vehicle's updated state is observed to compute the action reward. The specific steps of the fuzzy Q-learning algorithm are elucidated in the following section. For more details, readers are referred to Glorennec and Jouffe (1997).

2.2.3 Line tracking

The line tracking controller should be rather efficient and run at a high frequency such that the vehicle can stay on track, especially when it comes to winding tracks.

The most popular vehicle lateral movement controller (line tracking controller) in the existing literature is pure pursuit (Coulter, 1992). The input of pure pursuit is vehicle orientation error/tracking error e^o, which is the angle between the vehicle's orientation and the look-ahead line. The output is the adjusted orientation (turning angle) $\tilde{o}$, computed as Eq. (3):

$$\tilde{o}(t) = \arctan\left(\frac{2l \times \sin(e^o(t))}{d}\right) \tag{3}$$

where l is the vehicle length and d is the look-ahead distance.

Besides pure pursuit, other efficient methods can also be used for vehicle lateral control, such as PID control (Normey-Rico et al., 2001). The same control error as in pure pursuit or other error measurements can be used, e.g., the distance between the longitudinal centerline of the vehicle body and the track. The PID lateral control output is formulated as Eq. (4):

$$\tilde{o}(t) = k_p \times e^o(t) + k_i \times \int e^o(t)\,\mathrm{d}t + k_d \times \frac{\mathrm{d}e^o}{\mathrm{d}t} \tag{4}$$

where k_p is the proportional gain, k_i is the integral gain, k_d is the derivative gain, $e^o(t)$ is the tracking error, $\mathrm{d}e^o$ is the change in $e^o(t)$, and $\mathrm{d}t$ is the change in time. The lateral control output can be applied to the vehicle's steering system to adjust the orientation.

Fuzzy *Q*-learning algorithm.

- Step 0: Define learning rate η, discount factor ρ, exploration rate ϵ, and convergence threshold ∂.
- Step 1: Initialize the q-table with 0. $q(n,j)=0$, where $1\le n\le N$ and $1\le j\le J$, N is the number of state membership functions, and J is the number of action membership functions. Define the reward for each state $R_n=\begin{cases}-10^{|n-3|}, \text{if } |n-3|>0\\ 0 \quad\quad, \text{otherwise}\end{cases}$.
- Step 2: Observe the current state and choose an action for each fired rule. $a_n=\underset{j}{\operatorname{argmax}}\, q(n,j)_{1\le j\le J,1\le n\le N}$ with a probability of $1-\epsilon$; $a_n=\text{Unif}(1,J)$ with a probability of ϵ.
- Step 3: Calculate the control action a with the fuzzy controller. $a=\sum_{n\in\mathcal{N}} a_n\times\mu_n$, where n is the fired rule index and μ_n is the corresponding firing strength.
- Step 4: Calculate the Q-function value of the current state. $Q_t(s(t),a)=\sum_{n\in\mathcal{N}}\mu_n\times q(n,a_n)$.
- Step 5: Execute the action and observe the next state $s(t+1)$.
- Step 6: Calculate the Q-function value corresponding to the rule optimal actions. $V_t\big(s(t)\big)=\sum_{n\in\mathcal{N}}\mu_n\times\max\left(q(n,j)\right)$.
- Step 7: Calculate the reward of action a. $r(t+1)=\sum_{n\in\mathcal{N}}\mu_n\times R_n$, where R_n is the reward of state n.
- Step 8: Calculate the time difference. $\tilde{\varepsilon}=r(t+1)+\rho\times V_t\big(s(t)\big)-Q_t(s(t),a)$.
- Step 9: Update the q-table. $q(n,a_n)=q(n,a_n)+\eta\times\tilde{\varepsilon}\times\mu_n$.
- Step 10: End the algorithm when the q-table converges, i.e., the matrix distance between q-tables at two consecutive steps is less than ∂.

2.3 Benchmark

A benchmark is developed with two feedback loops for comparison based on the literature, as shown in Fig. 3. The longitudinal loop controls the vehicle's speed with a single-step controller (i.e., PID). The output is the adjusted DCV instead of the adjusted speed as the OLMPC. Therefore, the longitudinal control inner loop (RL agent) is no longer needed. The lateral loop controls the vehicle to follow the designated track with a tracking controller, like the OLMPC.

In the longitudinal control loop, the PID controller inputs are the location and speed control errors, e^x and e^v. The PID speed control output $\lambda(t)$ is formulated as Eq. (5):

$$\lambda(t)=K_p^{\lambda}\times e^x(t)+K_{p'}^{\lambda}\times e^v(t)+K_i^{\lambda}\times\int e^v(t)\,\mathrm{d}t+K_d^{\lambda}\times\frac{\mathrm{d}e^v}{\mathrm{d}t} \tag{5}$$

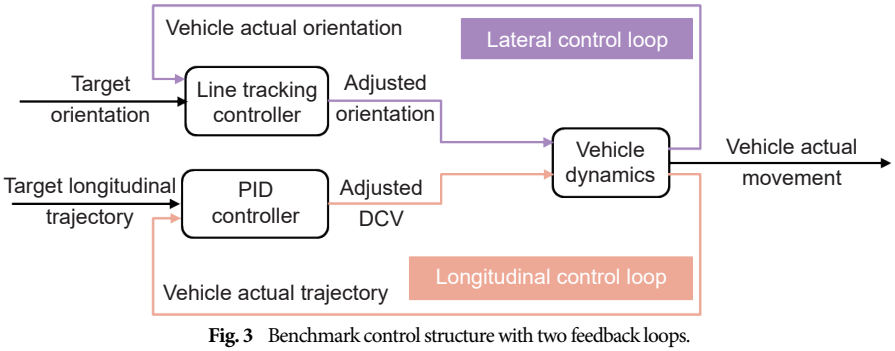

Fig. 3 Benchmark control structure with two feedback loops.

where K_p^λ and $K_{p'}^\lambda$ are the proportional gains, K_i^λ is the integral gain, K_d^λ is the derivative gain, and $\mathrm{d}e^v$ is the change in $e^v(t)$.

3 Reduced-scale platform

Given the limited resources, this subsection constructs a reduced-scale robot car platform to test the effectiveness of the proposed OLMPC. Small-scale robot cars can be totally under control and thus will not impose any safety concerns during experiments. They are effective alternatives in testing theoretical models before moving into full-scale vehicles. The reduced-scale platform includes Pololu Zumo 32U4 robots, infrared sensors for distance measurement, a PC for extensive computations (an Intel Core i5 processor, 8G RAM, and macOS), Wi-Fi chips for communication, and a ring test track.

3.1 Robots

The Pololu Zumo 32U4 robot (75:1 HP) is used to construct the physical testbed. It has a length of 9 cm and a width of 10 cm. The reason for choosing this robot is 4-fold. First, it is controlled by an Arduino microcontroller, which is convenient to program and compatible with various accessories. Second, it is loaded with line sensors that facilitate line tracking. Third, it features an extra serial port that can install a communication chip for information sharing with the computer. Fourth, two robot motors are equipped with encoders through which we can read the real-time position and speed of the robot.

3.2 Distance measurement

An infrared sensor (i.e., GP2Y0A51SK0F) is installed on the robot to measure distance with almost no measurement delay. Analog signals (i.e., the output voltage) returned by this sensor are converted into digital signals (i.e., the measured distance). A robot can measure the distance between itself and the preceding robot with this sensor. With the measured distance, the robot can act appropriately to avoid collisions with the help of a fine-tunned proportional-derivative (PD) controller.

When vehicles/robots are distant, their movements are regulated by the proposed OLMPC. When consecutive vehicles/robots are close (i.e., the car-following distance is less than a certain threshold), the PD controller of the following vehicle/robot will be activated to guarantee safety.

3.3 Wireless communication

A Wi-Fi chip (i.e., ESP8266) is installed on the robot for communication. The reason for choosing this chip is that it consumes less power and supports IP/TCP protocol. With this chip loaded, the robot can communicate with the PC via the Wi-Fi router. The PC generates vehicle time–specific trajectories and sends them to robots via the Wi-Fi router. On the other hand, the robot's real-time speed is read from encoders and sent back to the PC.

3.4 Test track

We set up a ring track with an inner radius of 110 cm and a track width of 5 cm, as shown in Fig. 4. Robot line sensors are activated to track lines based on reflectivity. The reflectivity of the whiteboard is greater than that of the black tape. Thus, robots stick to the ring track of the black tape.

3.5 OLMPC customization for robots without steering systems

The OLMPC proposed for full-scaled vehicles directly applies to robots with steering systems through which we can control the robot's turning angle. Yet, when it comes to robots without steering systems, e.g., Pololu Zumo 32U4 robots, the OLMPC needs to be modified. Specifically, the lateral control loop can no longer be independent of longitudinal loops but cascades inside, as shown in Fig. 5.

Without a steering system, the angular movement of the robot is accomplished by assigning different DCV values (i.e., rotation per minute) to two motors. Specifically, a greater DCV is given to the right motor if the robot needs to adjust to the left, and a greater DCV is assigned to the left motor if the robot needs to adjust to the right.

Given the limited computation capability of the Pololu Zumo robots, a PID controller is chosen for line tracking, given its satisfying accuracy and excellent computation efficiency. Besides, the robot can complete the PID control without communicating with the PC via a Wi-Fi network, so no communication lags exist.

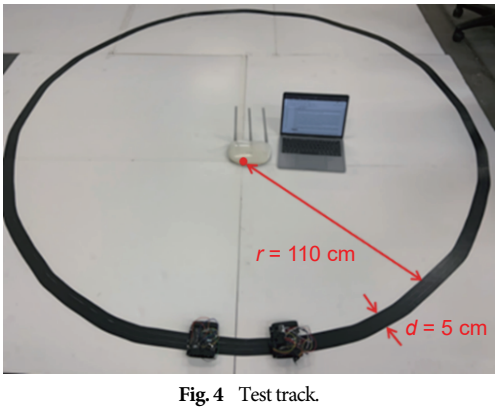

Fig. 4 Test track.

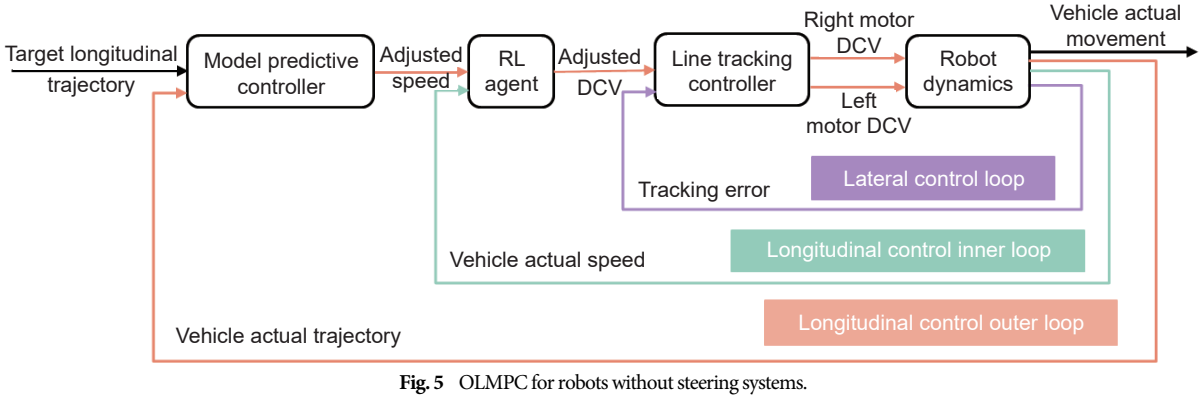

Fig. 5 OLMPC for robots without steering systems.

Thus, the PID lateral control is more efficient than other optimal control methods that must be done by the PC. The PID lateral control output $\tilde{o}(t)$ is formulated in Section 2.2.3.

The right motor DCV (DCV^R) and left motor DCV (DCV^L) are computed as Eq. (6) and Eq. (7), respectively:

$$DCV^R = DCV^{adj} - \tilde{o} \quad (6)$$

$$DCV^L = DCV^{adj} + \tilde{o} \quad (7)$$

where DCV^{adj} is the PID lateral control input, i.e., the adjusted DCV returned by the RL agent. It should be noted that if robots/vehicles with steering systems are used, the lateral control output can be directly used as the steering angle.

The lateral control loop runs every 0.05 s. The inner loop for longitudinal control operates every 0.1 s, while the other loop runs at a 0.2-s interval.

4 Experiments

This section conducts experiments to test the performance of the OLMPC. Section 4.1 tests the adaptivity of the OLMPC by changing the robot load. Next, Section 4.2.1 evaluates the OLMPC in regulating robot movements to follow time–specific reference trajectories to finish platoon formation and split operations. Section 4.2.2 compares the OLMPC with a benchmark in regulating robot movements.

4.1 Adaptivity tests

This study uses changes in the robot load as an example to investigate how the proposed OLMPC adapts to changes in the robot's operating environment. Objects are added on top of the robot to simulate load changes. The robot weighs about 275 g, including batteries. However, it can carry objects weighing much more than itself because of the great torque value, i.e., 23.3 cN·m. Two tests using two objects of different weights are conducted, respectively. The test results are presented in Fig. 6. The way objects are placed on robots is shown in the left corner of Fig. 6b.

In Fig. 6a, an object weighing 180 g is added on top of the robot at around 12 s. The robot slows down. The minimum speed is about 0.43 m/s (i.e., the dent in Fig. 6a). In response, the reinforcement learning agent automatically adjusts the conversion from speed to revolutions per minute (RPM) by updating the q-table to adapt to the new vehicle load. After 5 s of the load changing, the robot's speed is recovered to the target. In Fig. 6b, an object weighing 430 g is added to the robot at around 10 s. The robot speed drops to about 0.3 m/s. Similarly, the learning agent adjusts the conversion from speed to RPM to accommodate the load change. After 12 s of the load changing, the robot's speed is recovered to the target. More adapting time is needed than the first test because of the more substantial load change. Adapting time can be sped up by using more stable robots with better motors and employing advanced reinforcement learning techniques for more precise speed-to-RPM conversion.

In comparison, the load is also changed when the robot is controlled by the PID-based benchmark. In Fig. 6c, when adding the lighter object, the robot speed decreases to about 0.43 m/s. The PID speed controller tries to mitigate the speed difference from the target, as indicated by the slight speed increase after the dent (the red arrow). However, given the limited capabilities of the PID controller with fixed parameters, the robot speed is not recovered to the target. It stays around 0.47 m/s after. In Fig. 6d, the robot speed drops to only 0.29 m/s after adding the heavier object. The PID controller tries to make up the speed difference yet fails. The robot speed only recovered to about 0.32 m/s and stayed oscillating around it. To catch up with the target speed, the benchmark parameters (specifically, PID parameters) must be re-

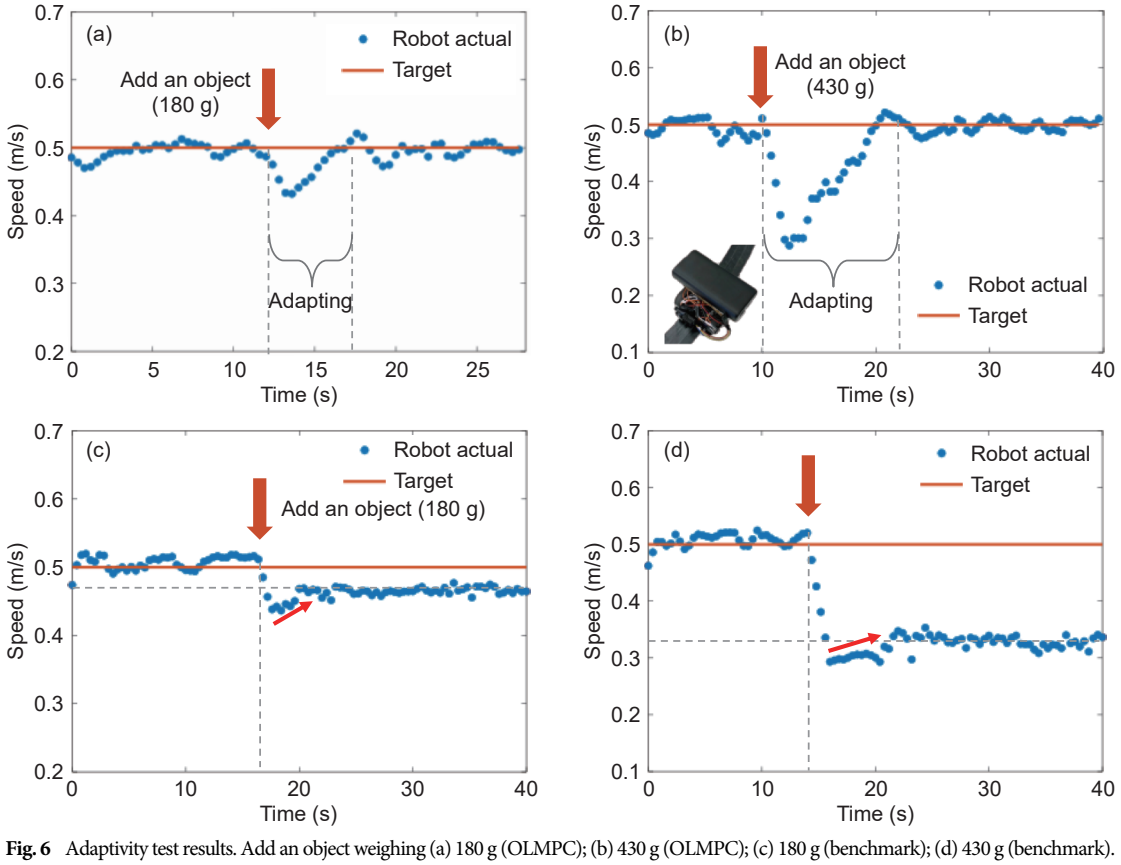

Fig. 6 Adaptivity test results. Add an object weighing (a) 180 g (OLMPC); (b) 430 g (OLMPC); (c) 180 g (benchmark); (d) 430 g (benchmark).

calibrated after the robot load changes. This requires additional effort and resources.

The above results indicate that the proposed OLMPC can automatically accommodate the changing operating environment to ensure the trajectory control performance, which greatly saves motor calibration resources (e.g., manually updating the speed to RPM conversion). Vehicle load change is only one of the common changes in the vehicle operating environment. Given the constraints of space, it is important to note that this study does not comprehensively test for variations in other factors like track grade and weather, which are also anticipated to occur frequently. This opens avenues for further research opportunities.

This subsection tests a constant target speed for illustration because the robot speed can quickly recover (in seconds). It should be noted that the OLMPC can also adapt to changes when the target is varying but with a longer adapting time. To facilitate future CAV time–specific trajectory control, state-of-the-art reinforcement learning convergence expedition techniques should be incorporated (Hossain et al., 2020; Majumdar et al., 2018; Nishio and Yamane, 2018).

4.2 Platoon operations

To test the effectiveness of the OLMPC in controlling robot motions, two fundamental platoon operations are tested, including platooning and split. Platooning is to cluster scattered vehicles/short platoons into long platoons such that vehicles can pass roadway segments more efficiently. On the opposite, split is to separate long platoons into individual vehicles/short platoons such that vehicles can reach their destinations. Detailed trajectory planning algorithms for the two operations can be found in Li and Li (2022, 2023). In this study, two robots are utilized to demonstrate the two operations. Key operation parameters follow. Platooning operation: maximum speed is 0.5 m/s, maximum acceleration is 0.02 m/s^2, minimum acceleration is -0.02 m/s^2, initial gap (or spacing) is 1.2 m, initial speed is 0.4 m/s, platooning gap is 0.18 m, and platooning speed is 0.5 m/s. Splitting operation: maximum speed is 0.5 m/s, maximum acceleration is 0.02 m/s^2, minimum acceleration is -0.02 m/s^2, initial platooning gap is 0.18 cm, initial platooning speed is 0.4 m/s, and ending gap is 1.2 m, and ending speed is 0.4 m/s. The platooning operation takes 12.4 s, and the split operation takes 10 s. Adaptivity is not tested during these two operations because of their short durations. The adapting time may be longer than the operation time. Future studies should incorporate state-of-the-art RL convergence expedition techniques to accelerate the adapting time. This study is intended to illustrate the feasibility of using RL methods to help control CAVs instead of proposing the perfect RL models.

4.2.1 *Online learning-based model predictive control*

Test results using the OLMPC are plotted in Fig. 7. The average root mean square error (RMSE) between the robot's actual trajectory and the planned reference trajectory is computed to quantify the control accuracy. It is shown that the two robots follow the planned trajectories to finish the platooning and split operations safely without much error.

In the platooning operation (Fig. 7a), the preceding robot's location RMSE is as small as 0.039 m, and the following robot's location RMSE is only 0.015 m. In the split operation (Fig. 7b), the preceding robot's location RMSE is 0.061 m, and the following robot's location RMSE is only 0.023 m. The small difference between the robot's actual trajectory and the planned trajectory demonstrates the effectiveness of the proposed OLMPC in regulating robot movements.

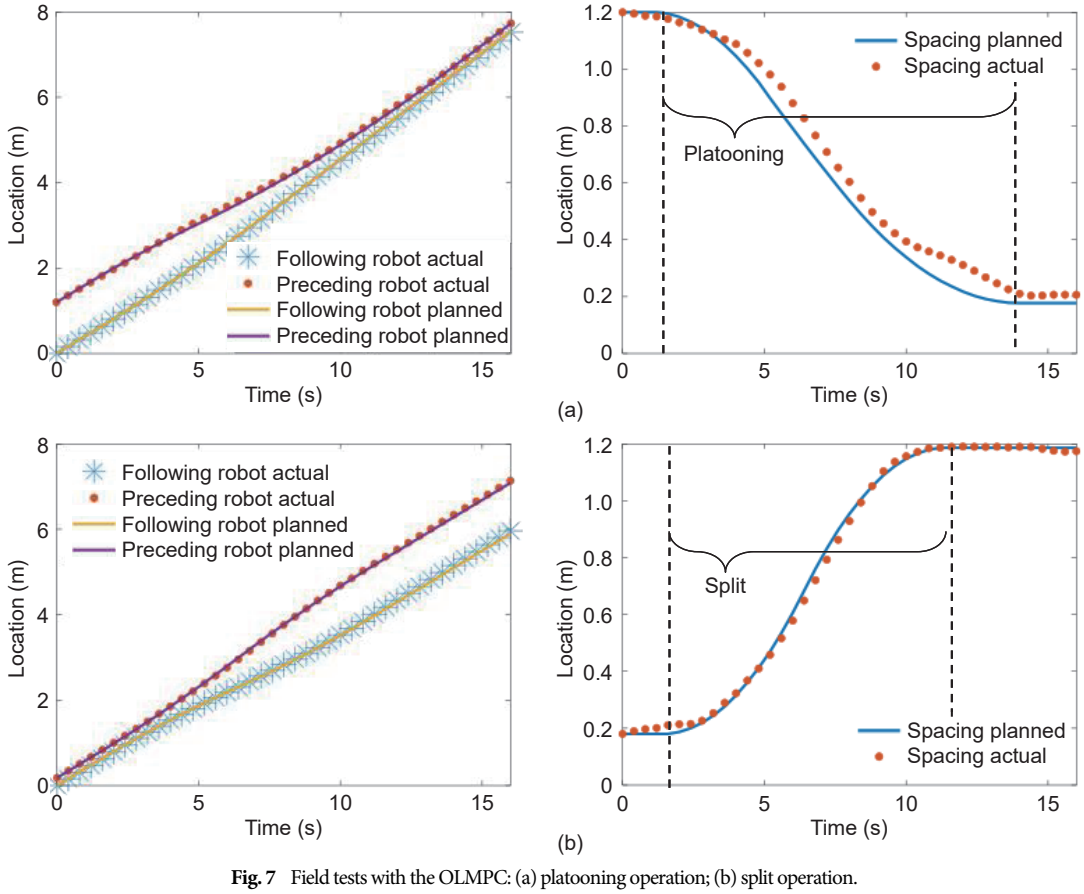

Fig. 7 Field tests with the OLMPC: (a) platooning operation; (b) split operation.

4.2.2 *Comparison with a benchmark*

This subsection compares the test results of the OLMPC with the benchmark by looking into the detailed control errors of a single vehicle.

The location and speed trajectories of the preceding vehicle when executing the platooning operation are plotted in Fig. 8. Fig. 8a shows the location control results of the OLMPC. The RMSE value is as small as 0.039 m. Fig. 8b shows the location

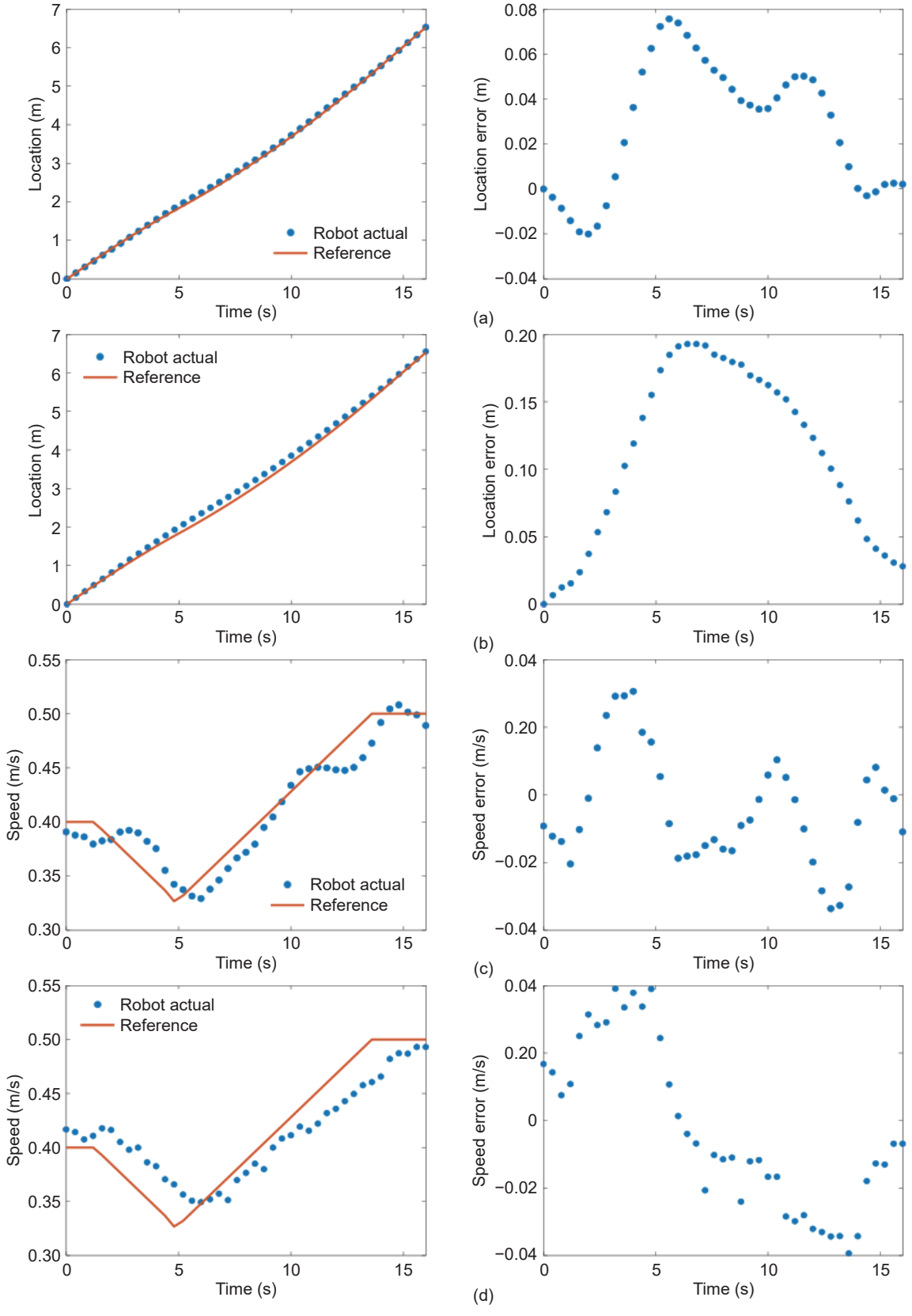

Fig. 8 Control result comparison: (a) OLMPC–location; (b) benchmark–location; (c) OLMPC–speed; (d) benchmark–speed.

control results of the benchmark (i.e., PID). The RMSE value is 0.126 m. The location control accuracy is improved by 69%. Fig. 8c shows the speed control results of the OLMPC. The RMSE value is as small as 0.017 m/s. Fig. 8d shows the speed control results of the benchmark (i.e., PID). The RMSE value is 0.024 m/s. The speed control accuracy is improved by 29%. The improvement illustrates the superiority of the proposed OLMPC in controlling robot time–specific trajectories compared with the benchmark.

5 Conclusions

This study proposes an online learning-based model predictive control (OLMPC) structure to regulate vehicle movements to follow time–specific trajectories. A model predictive controller is chosen to control the vehicle's longitudinal speed for higher accuracy than single-step controllers. Since the direct control variable of vehicles is throttle/brake (rotation per minute for robots) rather than speed (i.e., the output of the model predictive controller), a reinforcement learning agent is adopted to construct and maintain the dynamic conversion from the model predictive controller output to the vehicle/robot direct control variable. Operating environment changes can be automatically accounted for. This saves engine/motor parameter calibration resources and improves the trajectory control accuracy compared to constructing static lookup tables like the existing literature. Lateral movements are also controlled so vehicles/robots can operate on the designated track.

Reduced-scale robot car tests are conducted to verify the adaptivity of the OLMPC in the presence of operating environment changes (e.g., load change). Robot car tests are also carried out to finish a platooning operation and a split operation, which we propose in two previous trajectory planning studies. Results show that, with the OLMPC structure, robots' movements are regulated to follow the reference time–specific trajectories to finish the intended operations with much error. Specifically, in the platooning operation, the robot's location RMSE is as small as 0.039 m, and the speed RMSE is only 0.017 m/s; in the split operation, the robot's location RMSE is 0.061 m, and the speed RMSE is only 0.013 m/s. Further, the superiority of the OLMPC in regulating robot movements is demonstrated by comparing it with a benchmark (constructed based on PID). Overall, this study fills the void of time–specific CAV trajectory control, which is the premise of implementing theoretical trajectory planning models in the field and harvesting the benefits of the CAV technology. There are several research directions along which this study can be extended to. First, the control performance can be improved by upgrading the hardware. Specifically, line tracking can be more accurate using model prediction control if robots/vehicles with better computation capabilities and motor configurations are utilized. This will result in less disturbance on the longitudinal speed control, and the reinforcement learning agent can construct a more accurate speed to RPM/throttle/brake conversion. Currently, robot localization is accomplished by referring to the travel distance measured by the robot itself. Better localization accuracy is expected if a camera can be set up to overlook the test track. The same expectation applies to full-scale tests with CAVs equipped with Lidars that can accurately localize vehicles. Second, this study adopts a simple reinforcement learning method (i.e., *Q*-learning). More advanced models can be tested for a more accurate speed to RPM conversion. Also, expedition techniques should be incorporated to accelerate reinforcement learning convergence, especially when the target speed varies with time (Hossain et al., 2020; Majumdar et al., 2018).

Replication and data sharing

Code and data are accessible at https://github.com/CATS-Lab.

Acknowledgements

This research is sponsored by the National Science Foundation (CMMI #1558887 and CMMI #1932452).

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

References

Bolton, W., 2015. *Instrumentation and Control Systems*. Amsterdam: Elsevier, 281–302.

Campion, M., Ranganathan, P., Faruque, S., 2018. UAV swarm communication and control architectures: A review. J Unmanned Veh Sys, 7, 93–106.

Chiou, J. S., Tsai, S. H., Liu, M. T., 2012. A PSO-based adaptive fuzzy PID-controllers. Simul Model Pract Theory, 26, 49–59.

Clifton, J., Laber, E., 2020. *Q*-learning: Theory and applications. Annu Rev Stat Appl, 7, 279–301.

Coulter, C., 1992. Implementation of the pure pursuit path tracking algorithm. https://api.semanticscholar.org/CorpusID:62550799

Glorennec, P. Y., Jouffe, L., 1997. Fuzzy *Q*-learning. In: Proceedings of 6th International Fuzzy Systems Conference, 659–662.

He, X., Liu, H. X., Liu, X., 2015. Optimal vehicle speed trajectory on a signalized arterial with consideration of queue. Transp Res Part C Emerg Technol, 61, 106–120.

Hossain, M. A., Noor, R. M., Azzuhri, S. R., Z'aba, M. R., Ahmedy, I., Anjum, S. S., et al., 2020. Faster convergence of *Q*-learning in cognitive radio-VANET scenario. In: Advances in Electronics Engineering, Lecture Notes in Electrical Engineering, vol 619, 171–181.

Li, Q., Chen, Z., Li, X., 2022. A review of connected and automated vehicle platoon merging and splitting operations. IEEE Trans Intell Transport Syst, 23, 22790–22806.

Li, Q., Li, X., 2023. Trajectory optimization for autonomous modular vehicle or platooned autonomous vehicle split operations. Transp Res Part E Logist Transp Rev, 176, 103115.

Li, Q., Li, X., 2022. Trajectory planning for autonomous modular vehicle docking and autonomous vehicle platooning operations. Transp Res Part E Logist Transp Rev, 166, 102886.

Li, Q., Li, X., Huang, Z., Halkias, J., McHale, G., James, R., 2021. Simulation of mixed traffic with cooperative lane changes. Computer Aided Civil Eng, 37, 1978–1996.

Li, Q., Yao, H., 2022. Individual variable speed limit trajectory planning considering stochastic arriving patterns. Int J Coal Sci Technol, 9, 1–17.

Ma, J., Hu, J., Leslie, E., Zhou, F., Huang, P., Bared, J., 2019. An eco-drive experiment on rolling terrains for fuel consumption optimization with connected automated vehicles. Transp Res Part C Emerg Technol, 100, 125–141.

Ma, J., Leslie, E., Ghiasi, A., Huang, Z., Guo, Y., 2020. Empirical analysis of a freeway bundled connected-and-automated vehicle application using experimental data. J Transp Eng Part A Syst, 146, 4020034.

Majumdar, A., Benavidez, P., Jamshidi, M., 2018. Multi-agent exploration for faster and reliable deep *Q*-learning convergence in reinforcement learning. In: 2018 World Automation Congress (WAC), 1–6.

Milanes, V., Godoy, J., Villagra, J., Perez, J., 2010. Automated on-ramp merging system for congested traffic situations. IEEE Trans Intell Transp Syst, 12, 500–508.

Morales, A., Nijmeijer, H., 2016. Merging strategy for vehicles by applying cooperative tracking control. IEEE Trans Intell Transport Syst, 17, 3423–3433.

Nishio, D., Yamane, S., 2018. Faster deep Q-learning using neural episodic control. In: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), 486–491.

Normey-Rico, J. E., Alcalá, I., Gómez-Ortega, J., Camacho, E. F., 2001. Mobile robot path tracking using a robust PID controller. Contr Eng Pract, 9, 1209–1214.

Raboy, K., Ma, J., Leslie, E., Zhou, F., 2021. A proof-of-concept field experiment on cooperative lane change maneuvers using a prototype connected automated vehicle testing platform. J Intell Transp Syst, 25, 77–92.

Rajamani, R., Tan, H. S., Law, B. K., Zhang, W. B., 2000. Demonstration of integrated longitudinal and lateral control for the operation of automated vehicles in platoons. IEEE Trans Contr Syst Technol, 8, 695–708.

Su, J., Wu, J., Cheng, P., Chen, J., 2018. Autonomous vehicle control through the dynamics and controller learning. IEEE Trans Veh Technol, 67, 5650–5657.

Tang, L., Yan, F., Zou, B., Wang, K., Lv, C., 2020. An improved kinematic model predictive control for high-speed path tracking of autonomous vehicles. IEEE Access, 8, 51400–51413.

Yang, X., Yang, X., Deng, X., 2020. Horizontal trajectory control of stratospheric airships in wind field using Q-learning algorithm. Aerosp Sci Technol, 106, 106100.

Yao, H., Cui, J., Li, X., Wang, Y., An, S., 2018. A trajectory smoothing method at signalized intersection based on individualized variable speed limits with location optimization. Transp Res Part D Transp Environ, 62, 456–473.

Yao, H., Li, X., 2020. Decentralized control of connected automated vehicle trajectories in mixed traffic at an isolated signalized intersection. Transp Res Part C Emerg Technol, 121, 102846.

Qianwen Li is an Assistant Professor at the University of Georgia. She received her Ph.D. degree from the University of South Florida in 2022. Her research interests are traffic data analytics and traffic flow modeling.

Peng Zhang is a Ph.D. student at the University of Wisconsin–Madison. He received his M.S. degree from the University of South Florida in 2019. His research interest is connected autonomous vehicle.

Handong Yao is an Assistant Professor at the University of Georgia. He received his Ph.D. degree from Harbin Institute of Technology in 2020. His research interests are cyber physical systems and autonomous vehicle control.

Zhiwei Chen is an Assistant Professor at Drexel University. He received his Ph.D. degree from the University of South Florida in 2020. His research interests are public transit systems and autonomous vehicles.

Xiaopeng Li is a Professor at the University of Wisconsin–Madison. He received his Ph.D. degree from the University of Illinois at Urban-Champaign in 2011. His research interests are automated vehicle traffic control and connected & interdependent infrastructure systems.